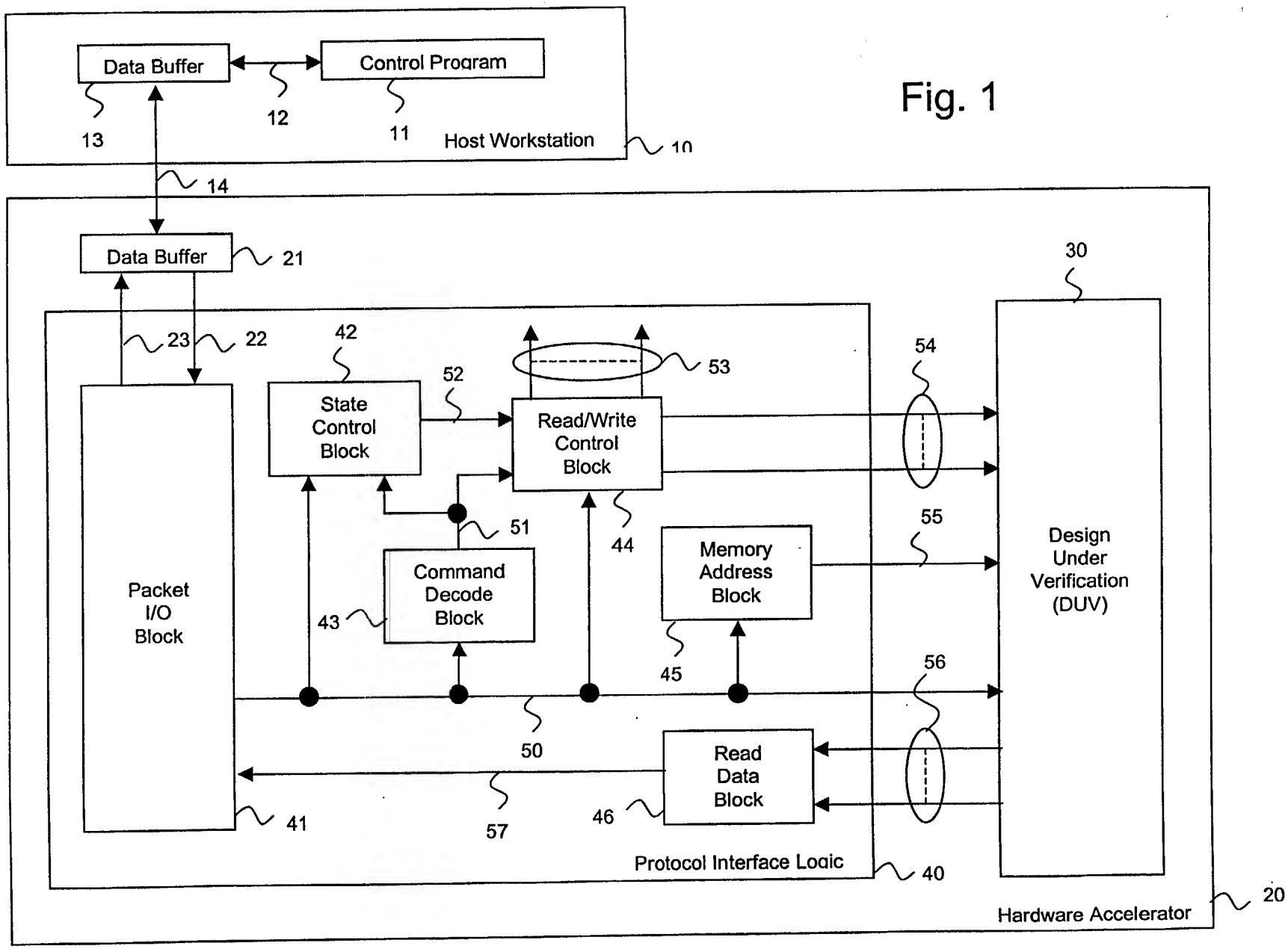

Fig. 1

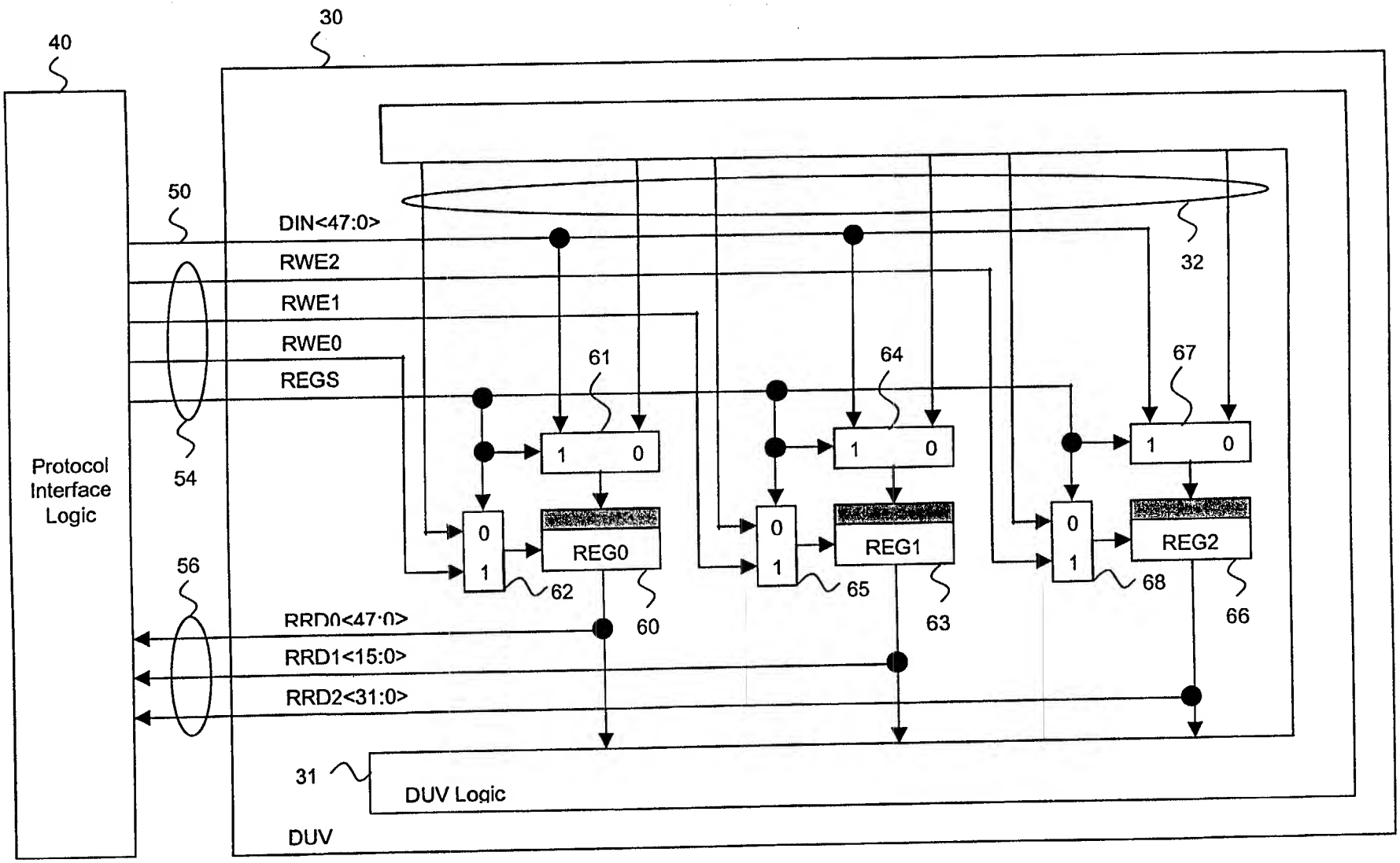

Fig. 2A

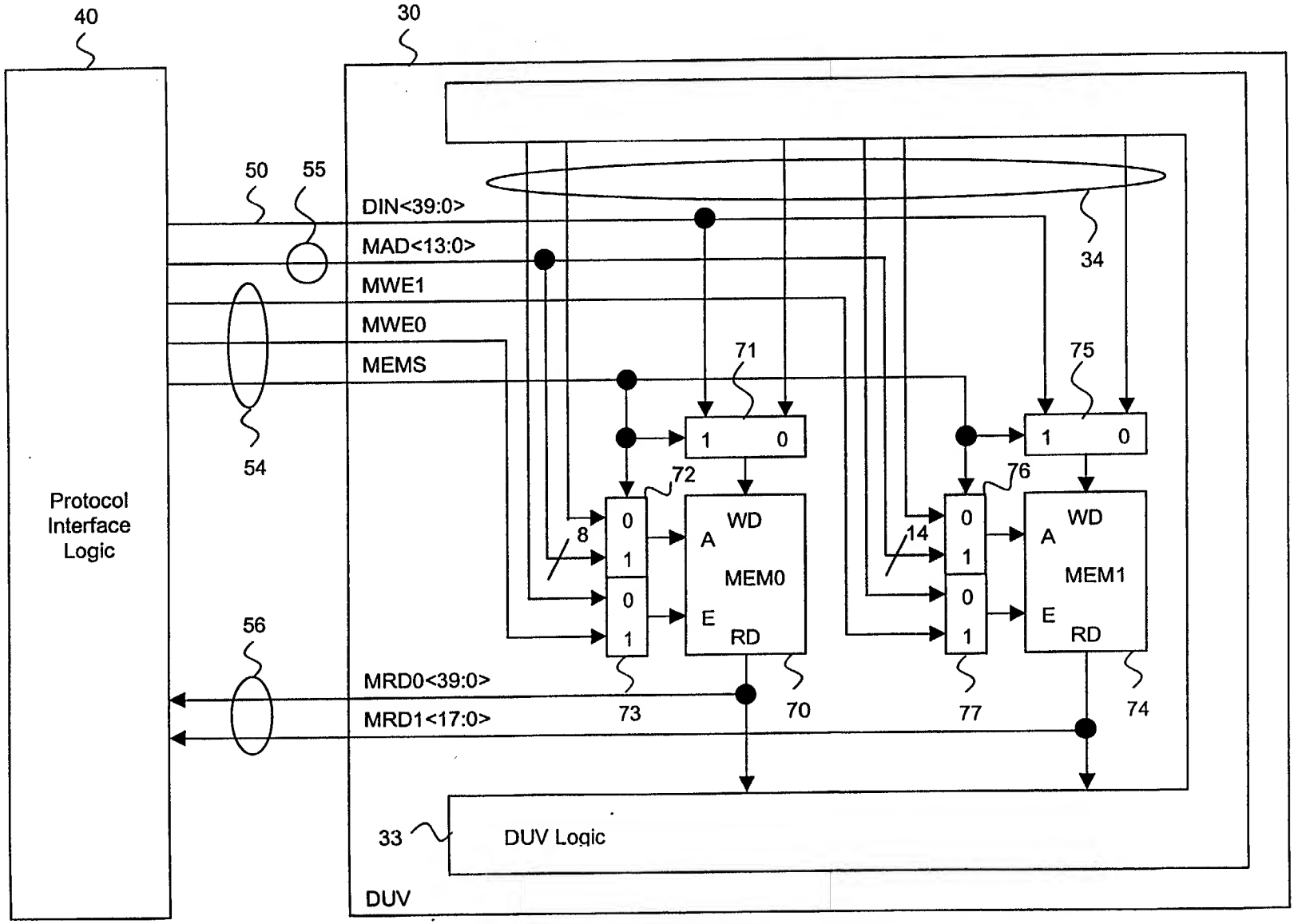

Fig. 2B

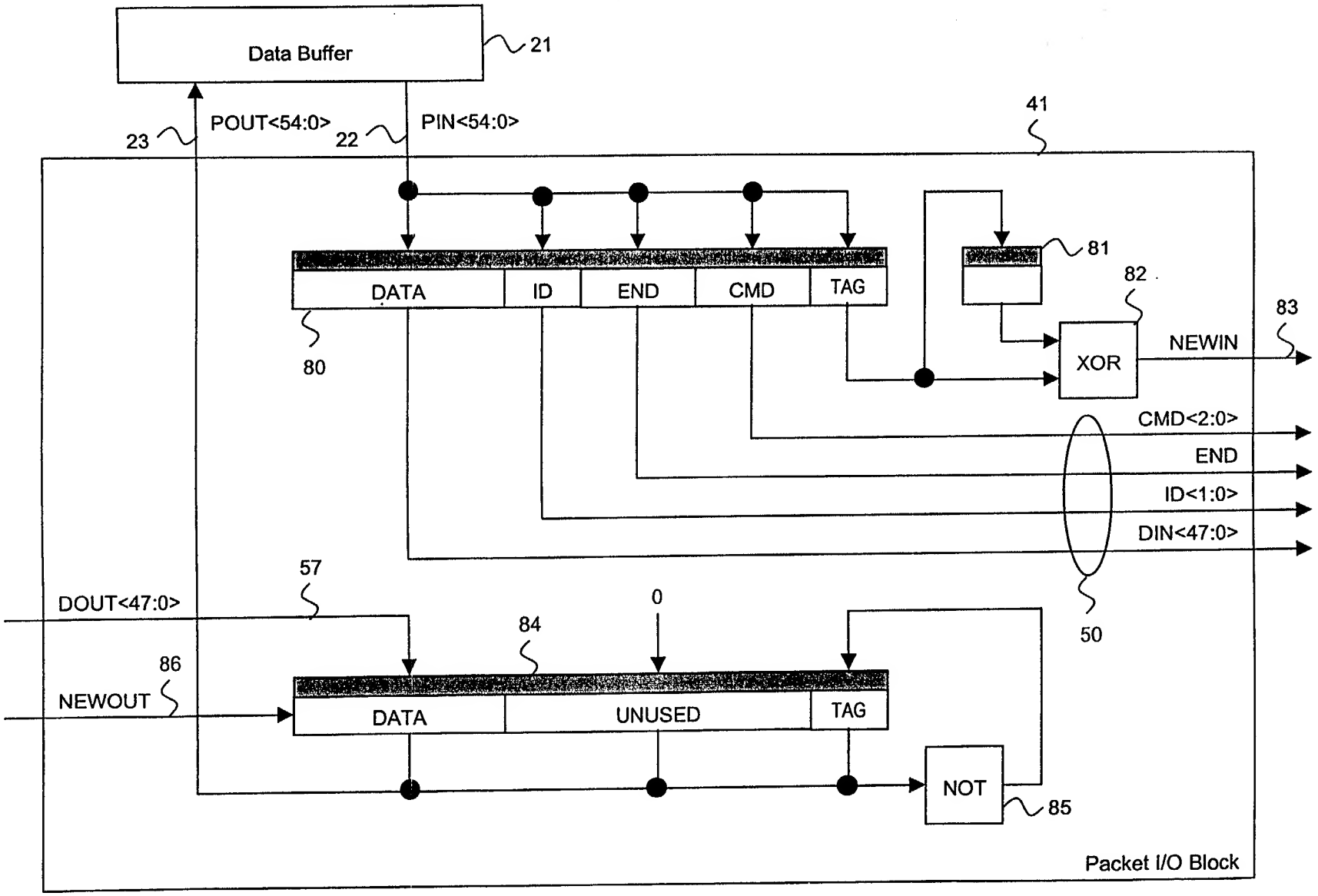

Fig. 3

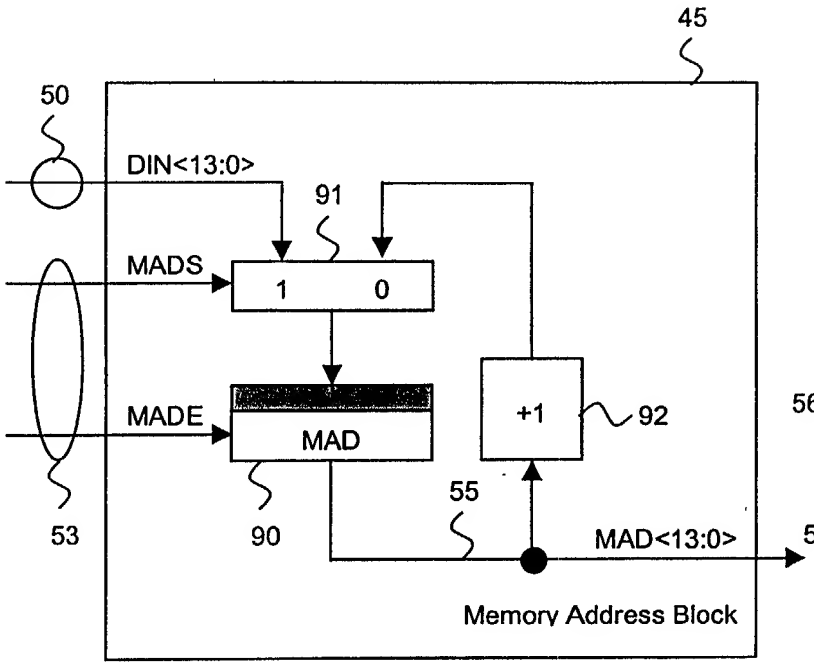

Fig.4

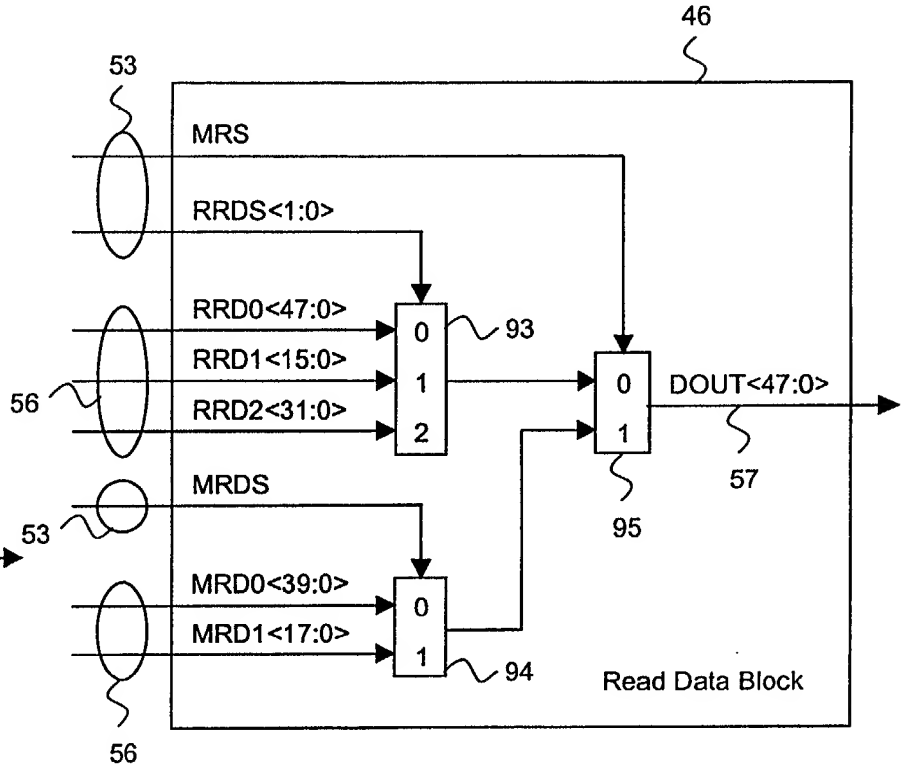

Fig. 5

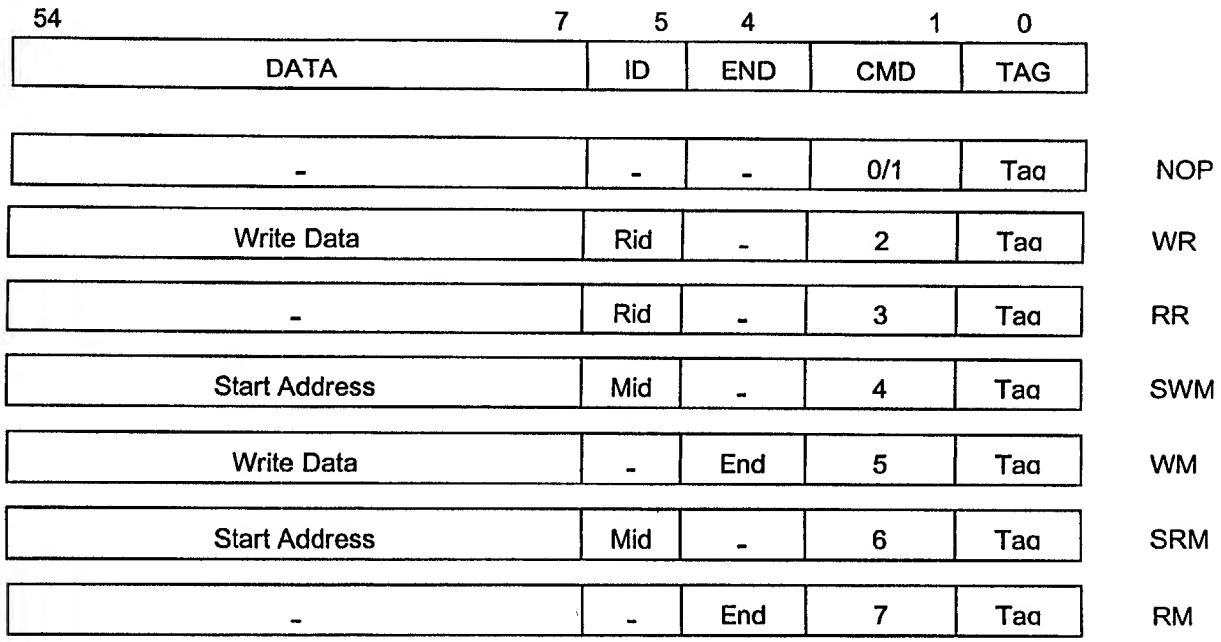

Fig. 6

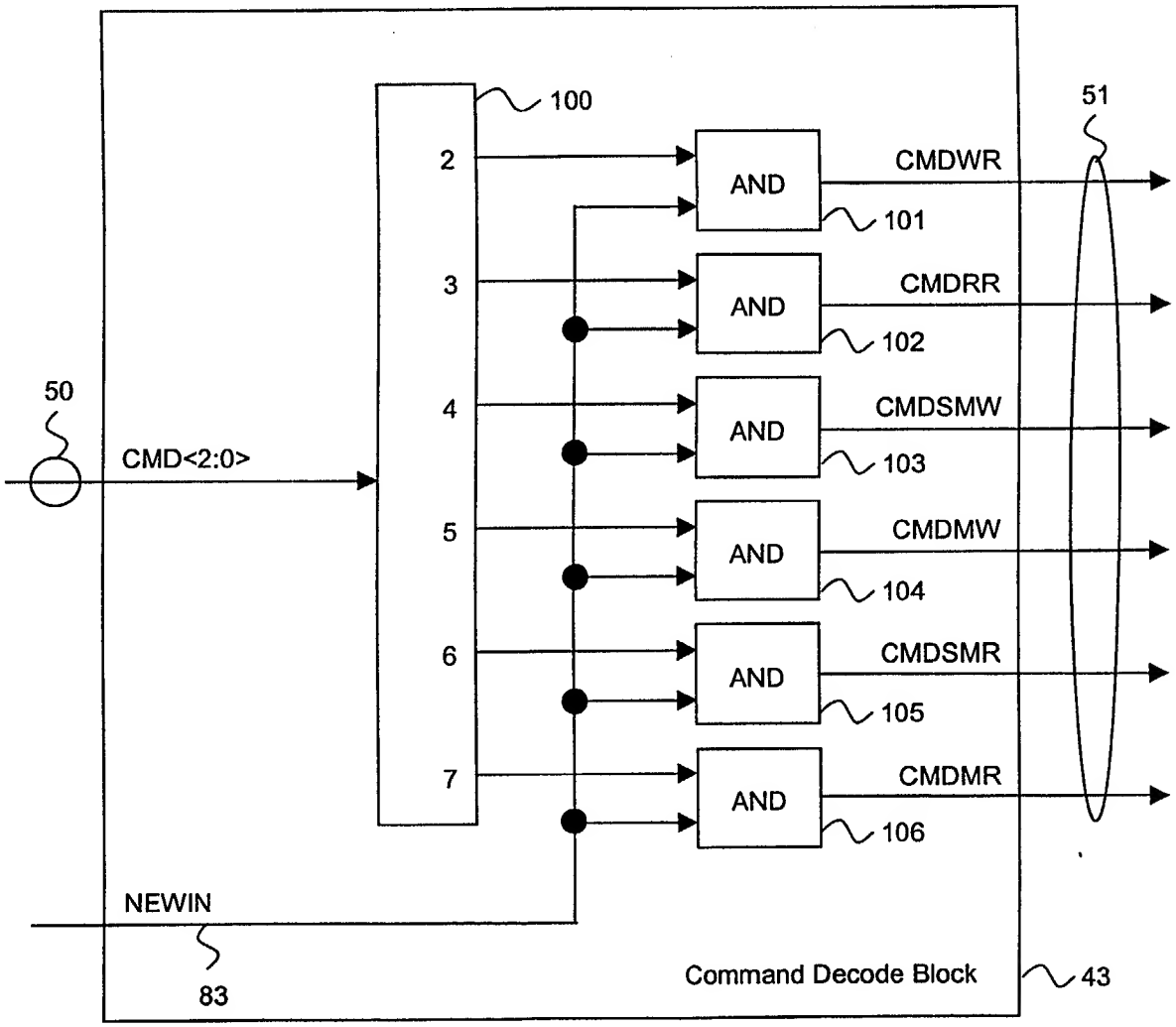

Fig.7

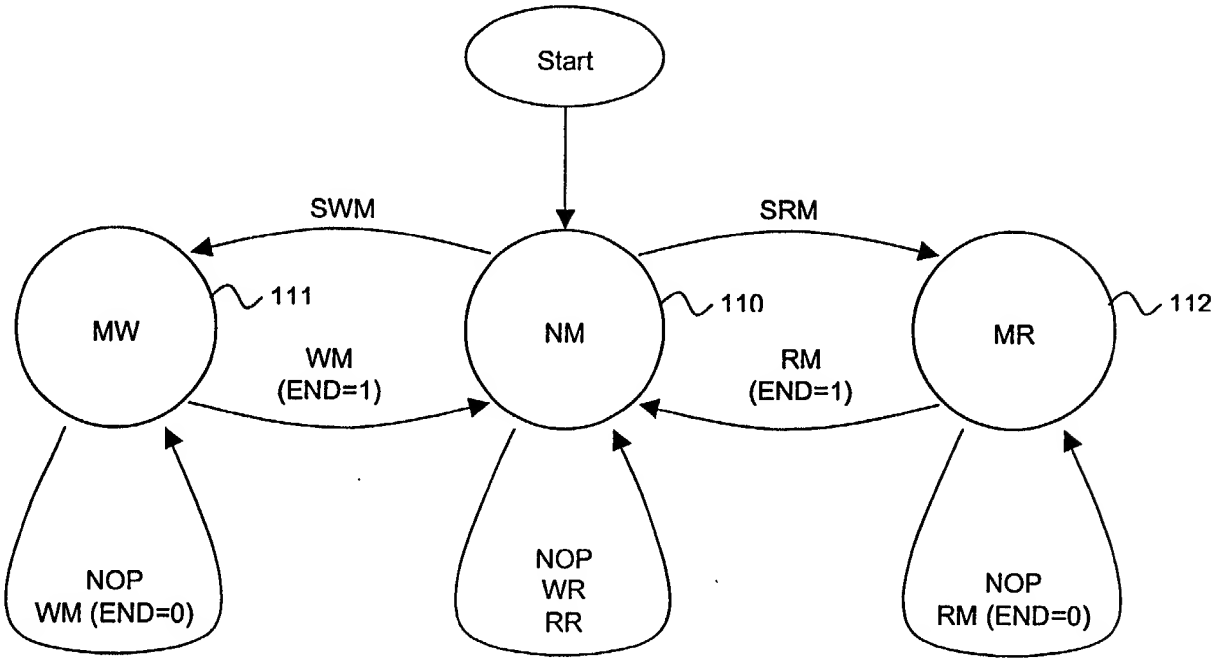

Fig. 8

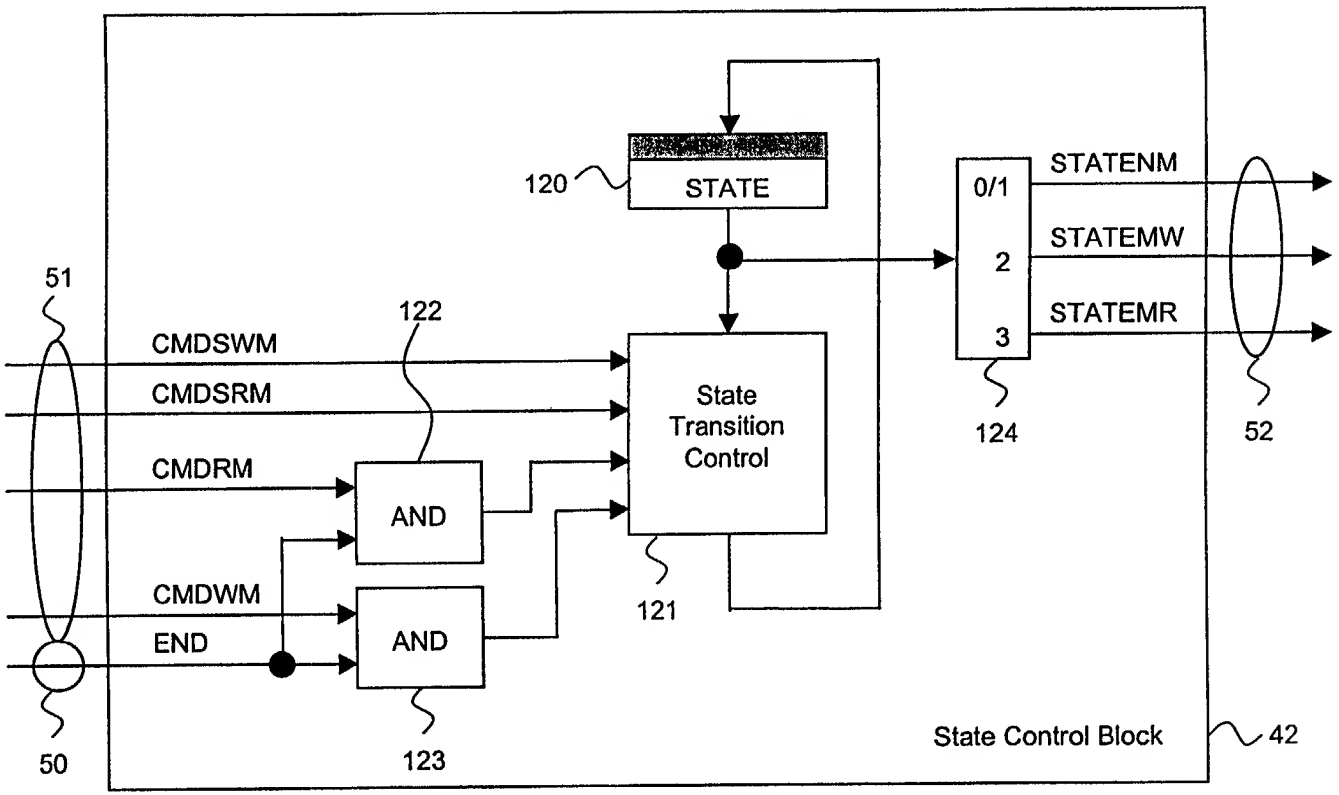

Fig. 9

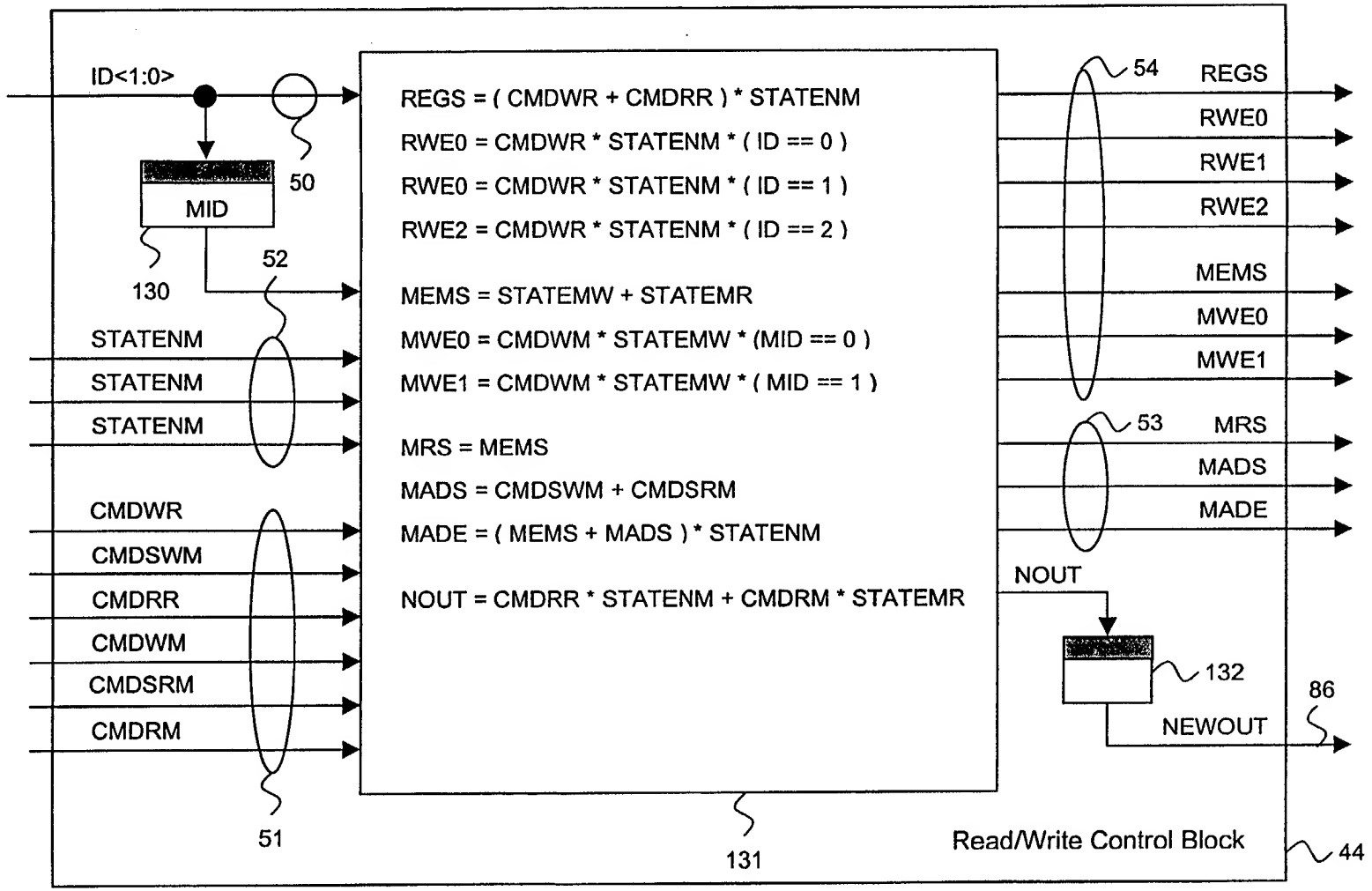

Fig.10

	CYCLE	N	N+1	N+2	N+3	N+4	N+5
140							
141	NEWIN	1	1	0	1	1	0
142	CMD	WR	WR	-	WR	WR	-
143	DIN	A	B	-	C	D	-
144	REG	-	A	B	-	C	D

Fig.11A

	CYCLE	N	N+1	N+2	N+3	N+4	N+5
140							
141	NEWIN	1	-	-	1	-	-
142	CMD	RR	-	-	RR	-	-
144	REG	A	-	-	B	-	-
145	POUT(DATA)	-	A	A	A	B	B
146	TRANSFER	-	-	A	-	-	B

Fig.11B

	CYCLE	N	N+1	N+2	N+3	N+4	N+5
140							
141	NEWIN	1	1	1	0	1	0
142	CMD	SWM	WM	WM	-	WM	-
143	DIN	10	A	B	-	C	-
147	MAD	-	10	11	12	12	13
148	MEM	-	A	B	-	C	-

Fig.11C

	CYCLE	N	N+1	N+2	N+3	N+4	N+5	N+6
140								
141	NEWIN	1	1	0	0	1	0	0
142	CMD	SRM	RM	-	-	RM	-	-
143	DIN	10	-	-	-	-	-	-
147	MAD	-	10	11	11	11	12	11
148	MEM	-	A	-	-	B	-	-
145	POUT(DATA)	-	-	A	A	A	B	B
146	TRANSFER	-	-	-	A	-	-	B

Fig.11D

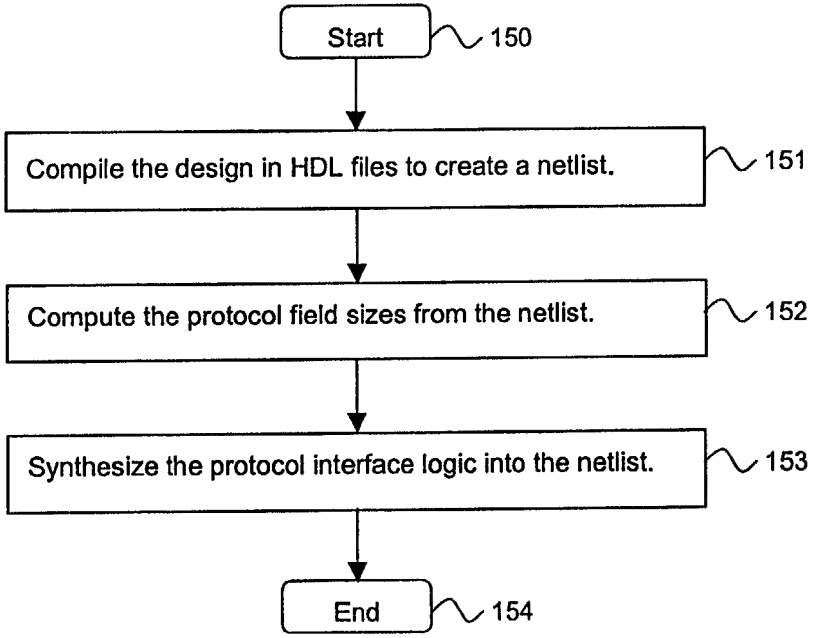

Fig.12

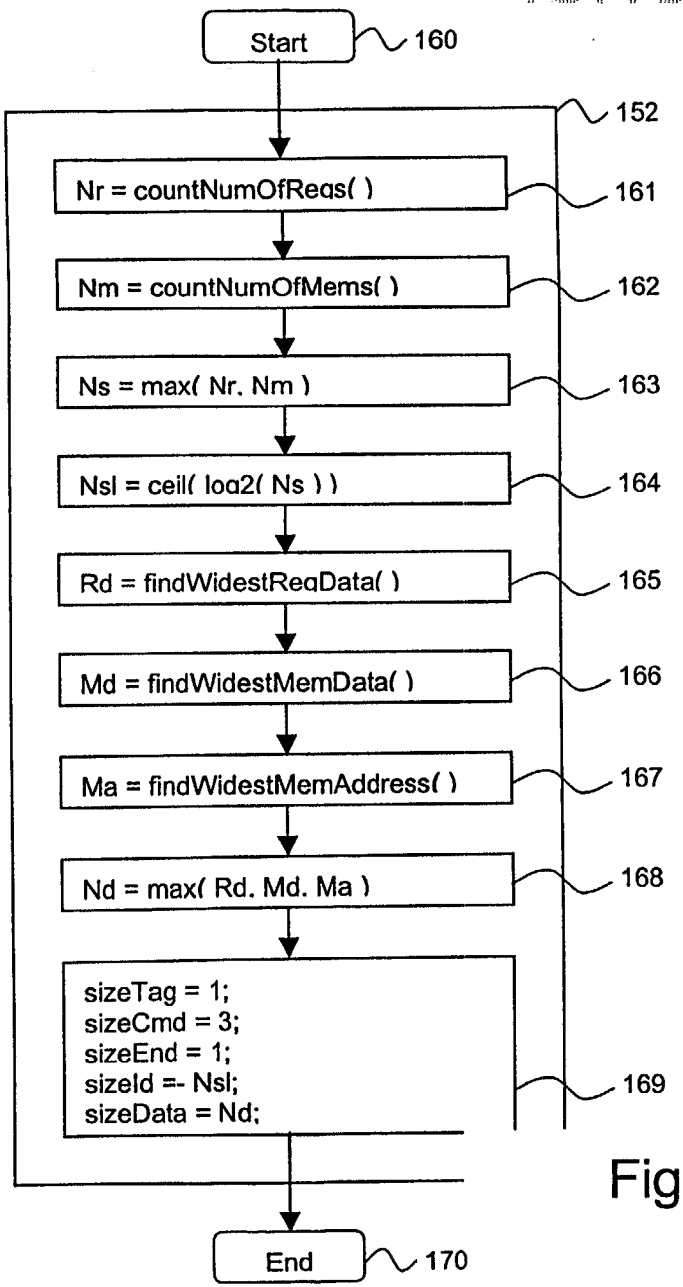

Fig. 13

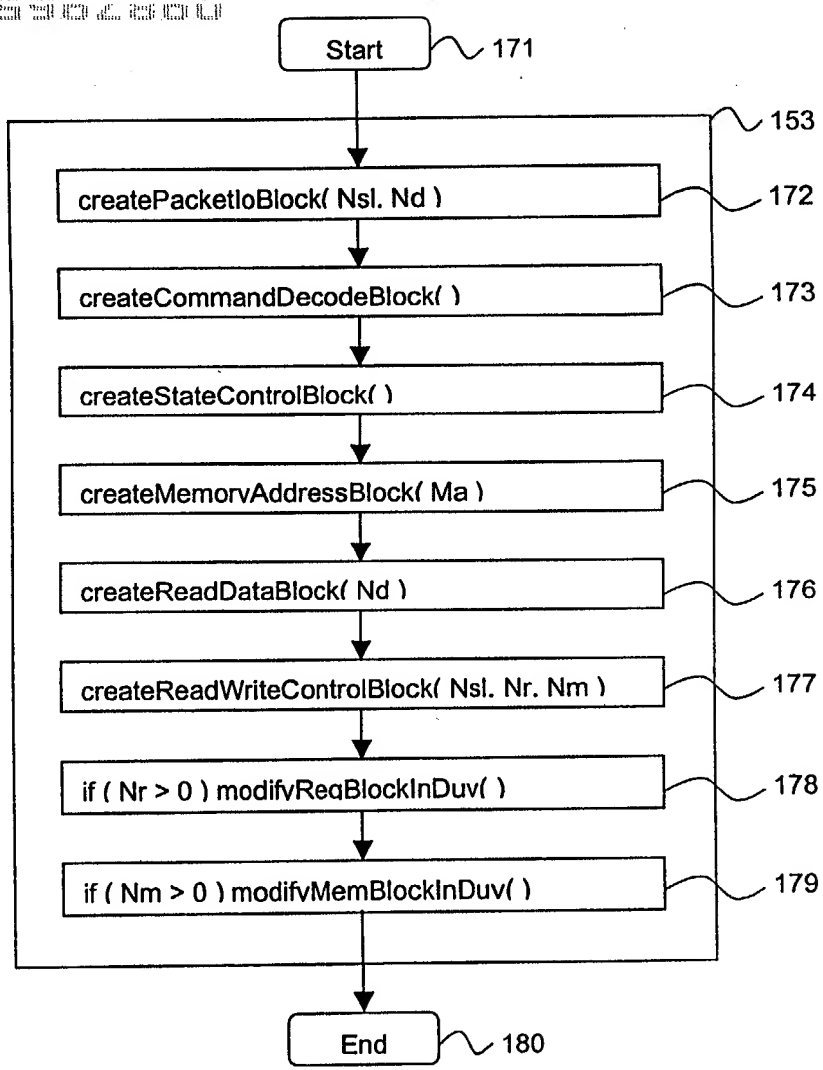

Fig.14

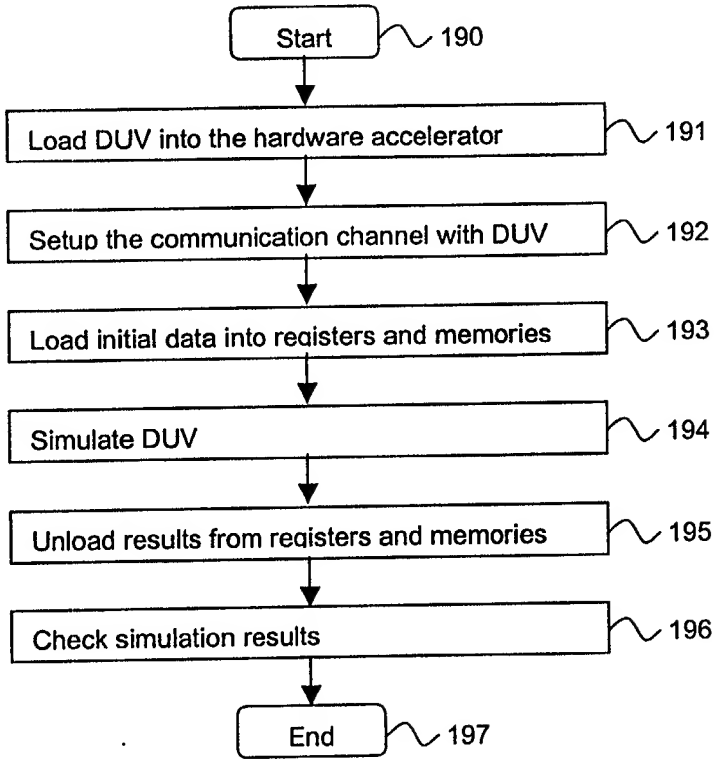

Fig.15

```
1   #define CMD_NOP 0  /* no operation         */
2   #define CMD_WR  2  /* write register       */
3   #define CMD_RR  3  /* read register        */
4   #define CMD_SWM 4  /* select write memory */
5   #define CMD_WM  5  /* write memory         */
6   #define CMD_SRM 6  /* select read memory  */
7   #define CMD_RM  7  /* read memory          */
8
9   #define uint unsigned int
10
11  int sizeId;   /* size of ID field in bits */
12  int sizeData; /* size of DATA field in bits */
13
14  uint tagOut = 0;  /* outgoing packet tag */
15  uint tagIn = 0;   /* incoming packet tag */
16
17  uint * packetOut;  /* outgoing packet */
18  uint * packetIn;   /* incoming packet */
19
20  extern void put_bits( uint *packet, int pos, int size, uint data );
21  extern void get_bits( uint *packet, int pos, int size, uint *data );
22  extern void copy_to_dbuff( int size, uint *packet );
23  extern void copy_from_dbuff( int size, uint *packet );
24  extern void transfer_to_haccel( int size );
25  extern void transfer_from_haccel( int size );
26
```

Fig.16

```
30  void sendPacket( uint cmd, uint end, uint id, int ds, uint data )
31  {
32      ps = 5 + sizeId + ds;
33      tagOut = tagOut ^ 1;
34      put_bits( packetOut, 0, 1, tagOut );
35      put_bits( packetOut, 1, 3, cmd );
36      put_bits( packetOut, 4, 1, end );
37      put_bits( packetOut, 5, sizeId, id );
38      put_bits( packetOut, 5+sizeId, ds, data );
39      copy_to_dbuff( ps, packetOut );
40      transfer_to_haccel( ps );
41  }
42
43  void receivePacket( int ds, uint *data )
44  {
45      ps = 5 + sizeId + ds;
46      while ( 1 ) {
47          transfer_from_haccel( ps );
48          copy_from_dbuff( ps, packetIn );
49          get_bits( packetIn, 0, 1, &tag );
50          if ( tag != tagIn ) break;
51      }
52      tagIn = tag;
53      get_bits( packetIn, 5+sizeId, ds, data );
54  }
55
```

Fig.17

```
60  void writeReg( uint rid, int ds, uint data )
61  {
62      sendPacket( CMD_WR, 0, rid, ds, data );
63  }
64
65  void readReg( uint rid, int ds, uint data )
66  {
67      sendPacket( CMD_RR, 0, rid, 0, NULL );
68      receivePacket( ds, &data );
69  }
70
71  void writeMem( uint mid, int as, uint start,
72                 int nw, int ds, uint *data )
73  {
74      sendPacket( CMD_SWM, 0, mid, as, start );
75      end = 0;
76      for ( i = 0; i < nw; i++ ) {
77          if ( i == (nw-1) ) end = 1;
78          sendPacket( CMD_WM, 0, end, ds, data[i] );
79      }
80  }
81
82  void readMem( uint mid, int as, uint start,
83                int nw, int ds, uint *data )
84  {
85      sendPacket( CMD_SRM, 0, mid, as, start );
86      end = 0;
87      for ( i = 0; i < nw; i++ ) {
88          if ( i == (nw-1) ) end = 1;
89          sendPacket( CMD_RM, 0, end, ds, data[i] );
90          receivePacket( ds, &data[i] );
91      }
92  }
```

Fig.18